DETAILED ACTION

1. Applicant's appeal brief dated December 21, 2009 has been fully considered. Examiner withdrew the previous Final action and the pending claims are being allowed.

EXAMINER'S AMENDMENT

2. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

3. Authorization for this examiner's amendment was given in a telephone interview by Mr. Brian C. Genco (Reg. No. 58,096) on October 26, 2010 and November 09, 2010 to obviate any potential 35 U.S.C 101 and/or 35 U.S.C 112, second paragraph issues and to place the claims in the condition for allowance.

4. In the interest of compact prosecution, examiner initiated an interview, a proposed amendments has been received via fax and being adopted by the examiner to amend claims 1, 4-5, 7, 9-19, 21-28, 30-32 and 35-38; cancel claims 2, 3, and 29; and further add new claim 39. See Examiner's Amendment below.

IN THE CLAIMS

Please amend claims 1, 4-5, 7, 9-19, 21-28, 30-32 and 35-38; cancel claims 2, 3, and 29; and further add new claim 39 as follows:

1. (Currently Amended) A system for non-intrusively monitoring an application, comprising:

~~at least one application stored on a computer-readable medium, the at least one application creates a shared memory area and stores application values in the shared memory area;~~

a first COBOL (Common Business Oriented Language) program module stored on a non-transitory computer-readable medium that is programmed to share[[s]] and attach[[es]] through a technical layer to [[the]]a shared memory area created by the application and ~~that is~~ used by the ~~at least one~~ application during real-time operation to store application values in the shared memory area, the first module is further programmed to read[[s]] at least one application value[[s]] from the shared memory area that [[have]]has been stored in the shared memory area by the ~~at least one~~ application during real-time operation, wherein the first COBOL program module is programmed with COBOL that does not provide support for features to share and attach to the shared memory area, and wherein the technical layer comprises a shared memory routine that includes functions for managing shared memory that enable

the first COBOL program module to share and attach to the shared memory area;

a second module stored on a non-transitory computer-readable medium in communication with the first COBOL program module that requests the first COBOL program module to read the application values, the second module receives the application values from the first module; and

a third module stored on a non-transitory computer-readable medium in communication with the second module that displays the application values.

2-3. (Canceled)

4. (Currently Amended) The system of Claim 1, wherein the at least one application value[[s]] is [[are]] further defined as at least one application variable and a value for the at least one application variable.

5. (Currently Amended) The system of Claim 1, wherein the first COBOL program module is further configured to communicate the application values to the second module in hypertext markup language format.

6. (Original) The system of Claim 1, wherein the third module is further defined as a graphical user interface.

7. (Currently Amended) The system of Claim 6, wherein the graphical user interface is further configured to receive an input identifying the application values to be read and configured to request the application values identified to the first COBOL program module, via the second module, and wherein the first COBOL program module is configured to read the requested application values data from the shared memory area and return the application variables to the graphical user interface, via the second module.

8. (Previously Presented) The system of Claim 6, wherein the graphical user interface is further configured to receive an input identifying requested application values to be displayed.

9. (Currently Amended) The system of Claim 1, wherein the first COBOL program module is further configured as a socket server through the technical layer and wherein the second module is further configured as a socket client such that the first COBOL program module and second module[[s]] communicate via a socket connection.

10. (Currently Amended) The system of Claim 1, wherein the first COBOL program module reads application values stored in the shared memory area by the at least one application while the at least one application is running.

11. (Currently Amended) The system of Claim 10, wherein the first COBOL program module reads application values stored in the memory area by the at least one application without interfering with the operation of the at least one application.

12. (Currently Amended) A method of non-intrusively monitoring operation of an application, comprising:

~~running an application in a real-time manner;~~

~~creating a memory area;~~

~~generating, by the application, application values during operation of the application;~~

~~writing, by the application, the application values in the memory area during the operation of the application;~~

attaching, by a COBOL (Common Business Oriented Language) program monitor through a technical layer, to a memory area created by the application, wherein the memory area includes application values generated by the application and written by the application to the memory area during operation of the application in a real-time manner, wherein the COBOL program monitor is programmed with COBOL that does not provide support for features to attach to the memory area, and wherein the technical layer comprises a memory routine that includes a function for attaching to a memory area that enables the COBOL program monitor to attach to the memory area;

reading, by [[a]]the COBOL program monitor, the memory area used by the application to obtain the application values, wherein at least one of the application values is not output by the application; and

displaying the application values read from the memory area.

13. (Currently Amended) The method of Claim 12, further comprising:

requesting, by a client, the application values from the COBOL program monitor; and

communicating the application values ~~variables~~ from the COBOL program monitor to the client.

14. (Currently Amended) The method of Claim 13, further comprising:

~~requesting application values;~~

~~running a plurality of applications in a real-time manner;~~

~~generating application values stored in one or more memory areas during operation of the plurality of applications;~~

attaching, by the COBOL program monitor through the technical layer, to a plurality of memory areas that include application values generated by a plurality of applications during operation of the plurality of applications in a real-time manner;

reading, by the COBOL program monitor, from the plurality of ~~the one or more~~ memory areas used by the plurality of applications to obtain

the application values generated by the plurality of applications;

and

displaying the ~~requested~~ application values generated by the plurality of applications.

15. (Currently Amended) The method of Claim 14, wherein the memory area is further defined as a block of memory ~~and wherein the monitor reads at least some of the application variables stored in the block of memory~~.

16. (Currently Amended) The method of Claim 13, further comprising providing a memory manager and wherein the COBOL program monitor registers with the memory manager to obtain a location of the memory area used by the application to store the application values.

17. (Currently Amended) The method of Claim 13, further comprising:

~~generating new application values by the application stored in the memory area, at least one of the new application values defined as a new value for a variable of the application;~~

requesting, by the client, that the monitor re-read the application values stored in the memory area, at least some of the application values including new application values generated by the application and stored in the memory area, at least one of the new application values defined as a new value for a variable of the application;

re-reading, by the COBOL program monitor, the memory area to obtain the new application values.

18. (Currently Amended) The method of Claim 17, wherein the COBOL program monitor reads the application values while the application is running.

19. (Currently Amended) The method of Claim 13, wherein the COBOL program monitor is configured as a socket server through the technical layer and wherein the client is configured as a socket client such that the communication between the COBOL program monitor and client is via a socket connection.

20. (Original) The method of Claim 12, wherein the application values are further defined as a variable of the application and a value of the variable.

21. (Currently Amended) A system for non-intrusively monitoring variables during operation of an application, comprising:

a compile listing stored on a non-transitory computer-readable medium having an address map with an offset associated with each of a plurality of variables of an application; and

a COBOL (Common Business Oriented Language) program module stored on a non-transitory computer-readable medium that is programmed to perform[[s]] reading of the compile listing and obtaining the offset of at least one of the plurality of variables of the

application, the COBOL program module further programmed to perform[[s]] attaching through a technical layer to an address space used by the application during real-time operation of the application to obtain a value for one or more of the plurality of variables written to the address space by the application during the real-time operation of the application using the offset, wherein the COBOL program module is programmed with COBOL that does not provide support for features to attach to the address space, and wherein the technical layer comprises a memory routine that includes a function for attaching to an address space that enables the COBOL program module to attach to the address space.

22. (Currently Amended) The system of Claim 21, wherein the COBOL program module is further ~~configured~~ programmed to read the compile listing and convert at least one of the plurality of variables to the associated offset.

23. (Currently Amended) The system of Claim 21, wherein the COBOL program module is further configured to search the compile listing and display the plurality of variables of the application for selection by a user.

24. (Currently Amended) The system of Claim 23, wherein the COBOL program module is responsive to selection by the user of one of the plurality of

variables to obtain the value for the selected one of the plurality of variables using the offset to locate the value of the variable in the address space.

25. (Currently Amended) The system of Claim 24, wherein the COBOL program module is further configured to display the selected one of the plurality of variables.

26. (Currently Amended) The system of Claim 21, wherein the address space is further defined as a memory space and wherein the COBOL program module attaches, using a socket layer of the technical layer, to the memory space used by the application.

27. (Currently Amended) The system of Claim 26, wherein the COBOL program module attaches, using the offset, to the memory space used by the application via an operating system service accessible through the technical layer.

28. (Currently Amended) The system of Claim 21, wherein the COBOL program monitor is further configured, using the compile listing, to query the address map for one or more of the plurality of variables of the application.

29. (Canceled)

30. (Currently Amended) The system of Claim 21, wherein the COBOL program module is further configured to attach to the memory space where the application is operating and overwrite the value for one or more of the plurality of variables using the offset.

31. (Currently Amended) The system of Claim 21, wherein the COBOL program module comprises:

a reader component ~~configured~~ programmed to ~~perform reading~~ read the compile listing and further ~~configured to perform converting~~ programmed to convert at least one of the plurality of variables of the application to the associated offset; and

a search component [[that]]programmed to receive ~~performs receiving~~ the associated offset of the at least one of the plurality of variables from the reader component, the search component ~~configured to perform attaching~~ programmed to attach to the application and ~~further operable to~~ locate the value of the at least one of the plurality of variables using the offset.

32. (Currently Amended) The system of Claim 21, further comprising a display component operably coupled to the COBOL program module to perform receiving the value for the one or more of the plurality of variables, the display component configured to perform displaying the value.

33. (Previously Presented) The system of Claim 32, wherein the display component is configured to employ the value to display a heartbeat.

34. (Previously Presented) The system of Claim 32, wherein the display component is configured to employ the value to display as a percentage complete.

35. (Currently Amended) A system for non-intrusively monitoring COBOL application values, the system comprising:

a COBOL (Common Business Oriented Language) program stored on a non-transitory computer-readable medium that is programmed to create[[s]] a shared memory area through a technical layer, generate[[s]] program values, and store[[s]] the program values in the shared memory area during real-time operation of the COBOL program, wherein the COBOL program is programmed with COBOL that does not provide support for features to create the shared memory area, and wherein the technical layer comprises a shared memory routine that includes a function for creating shared memory that enables the COBOL program module to create the shared memory area; and

a COBOL program monitor module stored on a non-transitory computer-readable medium that is programmed to share[[s]] the shared memory area with the COBOL program through the technical layer, and the COBOL program monitor module is programmed to read[[s]] the program values stored in the shared memory area by the COBOL program during real-time operation of the

COBOL program, wherein the COBOL program monitor is programmed with COBOL that does not provide support for features to share the shared memory area, and wherein the technical layer comprises a shared memory routine that includes a function for sharing shared memory that enables the COBOL program module to share the shared memory area.

36. (Currently Amended) The system of Claim 35, further comprising:

a second COBOL program configured to generate second program values and store the program values in the shared memory area during real-time operation of the second COBOL program, and wherein the COBOL program monitor module is further configured to read the second program values stored in the shared memory area by the second COBOL program.

37. (Currently Amended) The system of Claim 35, further comprising:

a second memory area; and

a second COBOL program configured to generate second program values and store the program values in the second memory area during real-time operation of the second COBOL program, and wherein the COBOL program monitor module is further configured to read the second program values stored in the second memory area by the second COBOL program.

38. (Currently Amended) The system of Claim 35, further comprising:

a user interface configured to monitor and display the application values; and

a client application in communication with the user interface and the COBOL program monitor module, the client application configured to request the program variables of the COBOL program from the COBOL program monitor module and provide the program variables to the user interface for display via the user interface responsive to a request from the user interface.

39. (New) The system of claim 1, wherein the second module is a second COBOL program module, wherein the second COBOL program module and the first COBOL program module are in communication using a socket connection established through the technical layer, wherein the first COBOL program module and the second COBOL program module are programmed with COBOL that does not provide support for socket connections, and wherein the technical layer comprises a socket routine that enables the first COBOL program module and the second COBOL program module to manage socket communications through operating system calls.

- **END OF AMENDMENT –**

Allowable Subject Matter

5. Claims 1, 4-28, and 30-39 (renumbered as 1-36) are allowed.

6. The following is an examiner's statement of reasons for allowance:

As Appellant pointed out in the Appeal Brief dated December 21, 2009, VII. ARGUMENT, pp. 23 – 26 – *an application that creates a shared memory area; the processes create the shared memory or memory blocks within the shared memory; shared memory and socket functionality are enabled for the COBOL applications through a technical layer that comprises shared memory and socket routines*, that prior arts of record taken alone or in combination does not disclose and/or fairly suggest such manners of claimed limitations as now recited (currently amended) in each of independent claims 1, 12, 21 and 35, and thus all pending claims are allowed.

7. Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Conclusion

8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is

(571) 270-1240. The examiner can normally be reached on 8:00-5:30 (EST/EDT), Monday through Friday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on (571) 272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Ben C Wang/
Examiner, Art Unit 2192

/Tuan Q. Dam/
Supervisory Patent Examiner, Art Unit 2192